DETERMINISTIC SERIAL BUS COMMUNICATION SYSTEM WITH "CONTROLLER AREA (CAN)" PROTOCOL

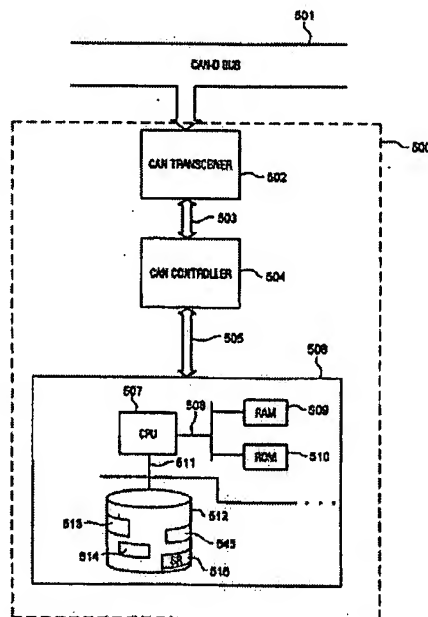|	
Patent number:	JP2002500841T
Publication date:	2002-01-08
Inventor:	
Applicant:	
Classification:	
- International:	H04L12/413; H04L12/417; H04L12/64; H04L12/407; H04L12/64; (IPC1-7): H04L12/40
- european:	H04L12/413B; H04L12/417
Application number:	JP19990500714T 19980513
Priority number(s):	US19970863531 19970527; WO1998US09730 19980513

Also published as:

WO9854872 (A1)
EP1002399 (A1)
US6111888 (A1)
EP1002399 (A0)
EP1002399 (B1)
CN1153413C (C)

less <<

Report a data error here

Abstract not available for JP2002500841T
Abstract of correspondent: WO9854872

An apparatus and method for deterministically communicating data between multiple nodes (500) in a fashion that is consistent with the Controller Area Network ("CAN") communications protocol. The system applies to multiple nodes that functional blocks within an operating system environment and to multiple nodes that are each connected to a serial bus (501). The system utilizes standard CAN error checking, bus arbitration and message formatting and therefore uses standard CAN controllers and transceivers. One node on the bus is selected as the master node. The master node issues a periodic synchronization signal (401, 411) which defines time divisions (TDn) within which the operations of each node and communications over the CAN bus are organized. Data, particularly real-time data, is transmitted between nodes on the CAN bus during a known time division. Standard CAN bus arbitration is used to ensure that real-time data is transmitted over the CAN bus prior to the transmission of non-real-time data. This ensures that real-time data is, if appropriate, transmitted during each time division.

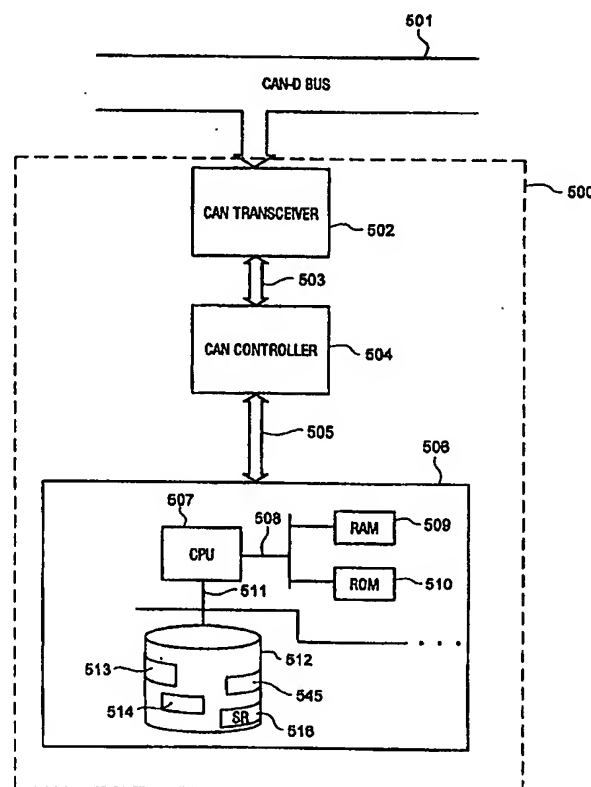Data supplied from the *esp@cenet* database - Worldwide

PCT

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: DETERMINISTIC SERIAL BUS COMMUNICATION SYSTEM WITH "CONTROLLER AREA (CAN)" PROTOCOL

(57) Abstract

An apparatus and method for deterministically communicating data between multiple nodes (500) in a fashion that is consistent with the Controller Area Network ("CAN") communications protocol. The system applies to multiple nodes that functional blocks within an operating system environment and to multiple nodes that are each connected to a serial bus (501). The system utilizes standard CAN error checking, bus arbitration and message formatting and therefore uses standard CAN controllers and transceivers. One node on the bus is selected as the master node. The master node issues a periodic synchronization signal (401, 411) which defines time divisions (TD$_n$) within which the operations of each node and communications over the CAN bus are organized. Data, particularly real–time data, is transmitted between nodes on the CAN bus during a known time division. Standard CAN bus arbitration is used to ensure that real–time data is transmitted over the CAN bus prior to the transmission of non–real–time data. This ensures that real–time data is, if appropriate, transmitted during each time division.

DETERMINISTIC SERIAL BUS COMMUNICATION SYSTEM WITH "CONTROLLER AREA (CAN)" PROTOCOL

FIELD OF THE INVENTION

The present invention generally relates to the field of serial bus communication systems. Specifically, the present invention relates to the field of serial bus communication systems using the Controller Area Network ("CAN") protocol and in particular to a method for adapting the CAN protocol to provide

5   deterministic communication of real time data between nodes on a communication bus and between functional blocks within a node.

STATEMENT OF THE PROBLEM

Data bus communication systems are used in countless applications. The control and manipulation of industrial processes, commonly referred to as the

10   process control industry, is exemplary of applications where the transmission of data between multiple nodes is required. The process control industry requires the communication of data between multiple sensors, e.g. flowmeters, and multiple control elements, e.g. valves. Sensors and controls elements are referred to herein collectively as instruments or instrumentation. Continuous processes are typically

15   monitored and controlled by measuring certain characteristics of a material being processed, inferring a state or condition of the process from the measured characteristics, and generating control signals which operate to control devices such as pumps, conveyer belts and valves to keep the process within a specified range of operation. The quality of a finished product, as well as the safety of the

20   process plant, is determined by the extent to which a process is properly controlled.

Some instrumentation in a process control plant produces data which is transmitted to a central control system. The control system processes the data from the various instruments and transmits control signals to control elements. A control element might be a valve, for example, which is opened and closed in

25   response to a control signal produced by the control system. All of the instrumentation and control systems in a given process plant are interconnected over some type of communications bus to allow the necessary processing of the measurement data and transmission of the control signals.

There are different types of data communicated over data busses. "Non-

30   real-time" data is data that is not rendered less valuable if the data is not received

as of a certain deadline. An example of non-real-time data is a request from a central control system to a pressure transmitter requesting the serial number of the pressure transmitter. Proper control of the system is not dependant on knowing with certainty the time at which the request was generated. "Real-time" data has

5 a specific deadline by which the data must be received or the data is lost or rendered useless. An example of real-time data is a mass flow rate signal transmitted from a mass flow instrument to a control system. The control system might, for example, use the mass flow rate signal to generate a control signal to a valve to maintain the mass flow rate through the system to within .5% of a target

10 value. In this case it is important that a continuous and predictable stream of mass flow data be delivered to the control system and from the control system to the control valve.

Since the current status of a process, as measured by its respective instrumentation, is processed and fed back to adjust the process itself, the real-time

15 nature of the process instrumentation data must be maintained. For this reason, among others, each piece of instrumentation in a system is typically separately connected to the control system and provides real-time data indicative of its respective measurement to the control system. The control system then provides the necessary timing and coordination to process multiple signals from multiple

20 instrumentation sensors to produce the necessary control signals. In some systems, each piece of instrumentation is provided a dedicated, hard-wired connection to the control system. This ensures that the data from each piece of instrumentation will be communicated to the control system in a way that preserves the real-time nature of the data. However, this system also requires a complex and

25 costly process plant wiring system in order to connect sometimes thousands of instruments to a control system.

It is sometimes the case that a single piece of instrumentation generates multiple data signals for purposes of local control or transmission to a control system. For example, a Coriolis mass flowmeter might be used to measure the

30 amount of fluid flowing into a tank. After a predetermined mass of the fluid has flowed into the tank, a valve is to turn off thereby stopping the flow of fluid into the tank. The data necessary to determine when the predetermined mass has been

2

reached includes the mass flow rate of the fluid and the total mass of the fluid flowed into the tank. Both of these values are provided by a Coriolis mass flowmeter. Rather than communicating a mass flow rate value and a total mass value from the Coriolis mass flowmeter to a control system for development of the

5   valve control signal, it is desirable to process the mass flow rate and total mass data and produce the valve control signal locally in the Coriolis mass flowmeter. In addition to the mass data, there are various non-real-time data signals processed within a Coriolis flowmeter. These signals include, but are not limited to, changes to the configuration data and tag information. These non-real-time signals are of

10   a lower priority than the real-time signals because a delay in processing them does not degrade the accuracy or performance of the flowmeter or the tank batching control. A data bus is sometimes necessary to coordinate the transmission of the various real-time and non-real-time data signals within a single instrument.

        Whether one considers an enterprise-wide communication bus across a

15   process plant or an intra-instrument bus within a single instrument computer, there are two primary types of communication protocols from which to choose. In Carrier-Sense Multiple Access ("CSMA") protocols, each node on the bus senses traffic and waits for traffic to clear before sending a message. When two nodes try to send messages at the same time, both nodes step back and try to transmit again

20   at a later time. Another category of communication protocols is known by the heading Time Division Multiple Access ("TDMA"). In TDMA-type protocols, each node on the bus is allotted a slot of time in which to transmit and receive messages. TDMA protocols typically are, among other things, deterministic. A deterministic communications protocol is one which provides ensured schedulability of the

25   actions in a real-time system.

        The Controller Area Network ("CAN") protocol is one protocol in the group of protocols known generally under the heading of CSMA. The CAN protocol has been adopted by the automotive industry as one of the standards for internal, operational data communications for automobiles. Typically, multiple sensors are

30   located throughout the vehicle at various locations. A two-wire serial bus is used to provide the communications between the multiple sensors and the microprocessor(s). For example, there may be sensors necessary for engine

management in the engine bay and sensors necessary for anti-lock brakes at or near each wheel. Output from these sensors is communicated over the two-wire bus to microprocessor(s) for use in determining the proper control of the vehicle. CAN provides, among other features, a means for arbitrating between the various

5 sensors to avoid conflicts between their respective signals. The arbitration includes a way to prioritize the signals so that more important signals are processed before less important signals. This allows the microprocessor to process a "brakes applied" signal before an "air-conditioning on" signal when both signals are initiated at the same time. CAN also provides a defined message format and an error

10 checking scheme for confirming the soundness of the communicated messages. CAN does not, however, provide a means to ensure that messages of a certain type or class, such as real-time messages, are communicated on a regular, deterministic basis.

Due to the adoption of the CAN protocol by the automotive industry,
15 microprocessors and peripherals specifically designed for implementing CAN-based systems are readily available. Their wide use and relative low cost makes these CAN processors an attractive choice for designers of communication buses.

There exists a need for a communication bus which provides the advantages of CSMA communication protocols such as arbitration, error checking and defined
20 message formats as well as the determinism provided by TDMA-like communication protocols. In particular, there exists a need for a deterministic CAN-based protocol which allows the designer to utilize the relatively inexpensive advantages of the CAN protocol and readily available hardware in a deterministic communications environment.

25   **STATEMENT OF THE SOLUTION**

The above identified problems, and others, are solved and a technical advance is achieved in the field by the deterministic serial bus communication system of the present invention. The present invention provides a method and apparatus by which standard CAN processors and related hardware are used in a
30 real-time, deterministic processing system. The advantages of the CAN protocol including inexpensive processors, message collision resolution, error handling and

message definition are combined with a time synchronization approach which provides ensured schedulability of the actions in a real-time system.

Generally, the present invention utilizes a synchronization signal transmitted over a serial bus to multiple nodes. The synchronization signal defines the start of

5   a time division for each of the nodes that are connected to the bus system. In one embodiment of the present invention, each node represents one or more sensors attached to a bus. For example, each piece of instrumentation in a chemical processing plant might be a node on the serial bus. In another embodiment of the present invention, each node represents one or more processing modules. For

10   example, a Coriolis flowmeter transmitter contains a mass flow processing module at one node and a density processing module at a second node. Both nodes are connected to the bus. In a third embodiment of the present invention, a node may contain multiple function blocks. For example, a Coriolis flowmeter transmitter contains a node which produces a batch control signal. The batch node contains

15   one function block that determines the totalized mass flow and a second function block that produces the batch control signal in response to the totalized mass flow amount. The intra-nodal communication of data between the function blocks within a node are conducted so as to be easily coordinated with the CAN protocol of the present invention. The CAN protocol of the present invention represents a hybrid

20   of the standard CAN protocol with the determinism of TDMA-based protocols. The communications protocol of the present invention is referred to herein as CAN with Determinism ("CAN-D").

In an embodiment of the present invention, each time division is comprised of three periods. During a first segment or period of the time division, a

25   synchronization signal ("sync signal") notifies all the nodes on the bus of the start of a new time division. Also during this period, each of the nodes on the bus generating real-time data queues up the appropriate data for transmission. During a second segment or period of the time division, each of the nodes generating real-time data transmits that data to the bus. During a third segment or period of the

30   time division, any node can request and transmit non-real-time data over the bus. The result is that the real-time data is schedulably transmitted to the bus during every time division. This ensured schedulability, or determinism, of the present

invention means that real-time data, used to control continuous, real-time processes, is transmitted at known, regular intervals over the bus.

In a further embodiment of the present invention, each sync signal initiates the next time division and concludes the previous time division. Messages from
5   nodes producing real-time data are given higher priority than messages from nodes producing non-real-time data. Standard CAN collision resolution operates to arbitrate message traffic over the bus. The length of the time divisions and the number of nodes on the bus are selected such that every real-time node can transmit a message during every time division. If there is not enough time
10   remaining during a time division after transmission of the real-time messages, then non-real-time messages are held over for transmission during the next time division.

In a further embodiment of the present invention, each synchronization signal initiates the next time division and concludes the previous time division. A
15   software router at each node directs the generation of messages for that node during the current time division. The messages at each node are generated in response to the directions from each node's respective software router. Real-time messages are queued during one time division for transmission at the start of the next time division. Non-real time messages are transmitted when time allows
20   during the current time division. With the next synchronization signal, the next time division begins. The queued real-time messages transmit at the start of the time division. After which the node software routers direct the next set of messages to be generated.

During each time division, the message arbitration, error checking and
25   message formatting is conducted according to the CAN protocol. The end of one time division and the start of the next time division is determined by the synchronization signal. If the bus is idle, meaning no messages are being transmitted, at the time the synchronization signal is initiated, then the next time division begins when the synchronization signal is initiated. If a multi-part message
30   is being transmitted over the bus at the time a synchronization signal is initiated, the synchronization signal may "cutoff" one or more CAN packets because of the higher priority of the synchronization signal. Any signal cutoff or terminated

because of the start of a new time division is transmitted in subsequent time divisions until it is successfully transmits in its entirety over the bus.

The present invention allows an instrumentation or plant designer to use the proven and inexpensive CAN protocol and CAN processors and to do so for the

5   processing of real-time data using a deterministic serial communications bus. The present invention also allows the modular design of instrumentation using standard CAN processors. The various processing modules of an instrument according to the present invention utilize the CAN-D protocol to communicate among one another. The present invention is also used to control intra-node communications

10 between functional blocks within a node.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a communications bus implementing the CAN protocol;

FIG. 2 is a timeline illustrating bus message activity in a communications

15 system according to the present invention;

FIG. 3 is a family of timelines illustrating bus message activity as well as node message generation activity according to the present invention;

FIG. 4 is a timeline illustrating a further embodiment of the communications system of the present invention;

20 FIG. 5 depicts a single node in greater detail;

FIG. 6 is a flowchart illustrating the processing steps related to communication of messages relative to a sync signal.

DETAILED DESCRIPTION

Controller Area Network (CAN) Protocol and Network - FIG. 1

25 FIG. 1 illustrates a network 100 including nodes 101-103. Each of nodes 101-103 is connected to two-wire bus 104. FIG. 1 shows only a section of bus 104 which extends in both directions.

Network 100 is typical of the networks of nodes in which the CAN protocol is utilized. An exemplary application is in the automotive industry where node 101

30 is an engine management control device, node 102 is a anti-lock brake sensor and node 103 is an anti-lock brake control device. Each node contains a CAN controller which is either embedded in a microprocessor or is a stand-alone controller. An

7

example of a microprocessor with an available embedded CAN controller is the Motorola MC68HC05 microprocessor. An example of a stand-alone CAN controller is the Siemens 81C90/91 - Stand Alone Full CAN Controller. The CAN controller is responsible for implementing the rules of the CAN protocol, discussed in more

5   detail below. Each node also contains a CAN transceiver which controls the physical interface between a node and the CAN bus. The most common physical interface, and the one used in the preferred embodiment of the present invention, is based on ISO 11898; "Road Vehicles - Interchange of digital information - Controller Area Network (CAN) for high speed communications." An example of a

10   CAN transceiver chip that is compatible with the ISO 11898 standard is the Phillips 82C250 CAN transceiver.

Node 101 includes CAN transceiver 105. CAN transceiver 105 communicates over intra-node path 108 with CAN controller 106. CAN controller communicates over intra-node path 109 with microprocessor 107. Each time

15   microprocessor 107 sends a message over bus 104, the message is actually sent from microprocessor 107 to CAN controller 106. CAN controller 106 is responsible for performing error checking, bus arbitration and message transmission all according to the CAN protocol. CAN controller 106 is physically connected to bus 104 through CAN transceiver 105. Likewise, node 102 includes CAN transceiver

20   110. CAN transceiver 110 communicates over intra-node path 113 with CAN controller 111. CAN controller communicates over intra-node path 114 with microprocessor 112. Node 103 includes similar functionality to nodes 101-102 but a different hardware configuration. The CAN controller is embedded in microprocessor 116. Communications between microprocessor 116 and CAN

25   transceiver 115 occur over intra-node path 117.

Applying the anti-lock brake system example to network 100, a sensor (not shown) inputs the wheel speed to microprocessor 112 of node 102. Node 102 operates to transmit a message over bus 104 to node 101 which includes a measure of the wheel speed sensed by node 101. Microprocessor 107 of node

30   101 calculates the appropriate actions in response to the message from node 102. Node 101 operates to communicate over bus 104 to node 103 a message directing the application of the brakes at the wheel controlled by node 103. Microprocessor

116 of node 103 outputs a signal to a brake control (not shown) to apply the brakes. All messages over bus 104 occur according to the CAN protocol.

CAN controllers 106, 111 and 116 coordinate the transmission of messages over bus 104. The CAN controllers implement the CAN message formatting, error
5   checking and message arbitration of the CAN protocol. Each message transmitted over bus 104 conforms to the CAN message format, described below. Message arbitration is necessary since multiple nodes are connected to bus 104. CAN provides an arbitration system that allows the prioritization of messages from different nodes. For example, if node 102, the anti-lock brake sensor device,
10   attempts to transmit a message over bus 104 at the same time that an engine management node (not shown) attempts to transmit a message over bus 104, the CAN protocol automatically determines which node gets to "talk" first over bus 104. In this example, anti-lock brake-related messages from node 102 might be deemed more important than engine management control-related messages and therefore
15   messages from node 102 are given a higher priority under the CAN protocol.

The CAN protocol (Version 1.2) utilizes an eleven bit identifier field in each message as a priority code to identify a particular device connected to bus 104. The identifier field is similar to a header and is followed by data fields. As is understood by those skilled in the art, the eleven bit identifier field of the CAN
20   protocol can be used to identify a particular device in the system or, alternatively, can be used to identify a particular message. All messages are prioritized with a message having the highest priority being assigned an identifier field having the smallest digital value. The next to the highest priority message is assigned the next to the smallest digital value, etc., down to the message having the lowest priority,
25   which is assigned the largest digital value. Transmit logic of each node 101-103 makes a bit by bit comparison between the message it is transmitting and the bus logic level. When two nodes happen to transmit on bus 104 simultaneously the node having the lower priority will have a recessive bit, e.g., a logic one, earlier in the message identifier, which will be over-written by a dominant bit, e.g., a logic
30   zero, of the higher priority message. Upon detecting this, the node transmitting the lower priority message will stop transmitting and become a "listener". This is the basic concept of CAN priority arbitration. Every node is assigned a different priority

(or prioritized) by virtue of its node address(es). The priority of each message is determined by the values set in the identifier field. The identifier field is the first field of every message sent by a node. The CAN identifier field and CAN arbitration in general is well known to those skilled in the art. The CAN protocol itself was

5   defined by Robert Bosch GmbH as "CAN Specification Version 1.2".

A standard component of CAN controllers and the CAN protocol is an error checking function. CAN error checking is typically done by the CAN controller and is a 16 bit, Cyclic Redundancy Check (CRC) checksum test.

Existing CAN systems are quite flexible and efficient to implement in part

10   due to the formatting, error checking and message arbitration described above. However, as noted, existing CAN systems offer no mechanism for ensuring that certain events or messages occur during a certain time period. If one is measuring, for example, the mass flow rate of a fluid through a pipe in grams per second, one must have certain information on a periodic basis in order to produce an accurate

15   measurement. Existing CAN systems cannot guarantee that certain information will be transmitted over the bus on a periodic basis. i.e., existing CAN systems are not deterministic. Existing CAN systems can only guarantee that a message of higher priority will be transmitted over the bus before a message of lower priority.

CAN with Determinism - In General - FIG. 2

20   FIG. 2 includes timing diagram 200 illustrating, in general, the operation of the present invention. FIG. 1 is also referred to with respect to the present invention since the basic hardware arrangement with respect to bus 104 is the same for networks according to the present invention as for networks according to existing CAN systems. The communications system of the present invention is

25   referred to herein as CAN with Determinism ("CAN-D"). Timeline 200 represents message activity over bus 104 during a sequence of time. A CAN-D communications system imposes a system time by transmitting sync pulses 201- 204 to produce equal, periodic time divisions TD_n, TD_{n+1} and TD_{n+2}. Sync pulses 201-204 are themselves CAN messages having high priority. Timeline 200

30   illustrates only several time divisions. One skilled in the art recognizes that this represents only a partial segment of the total system time.

Time division TD_n begins with the transmission of sync pulse 201 over bus 104. Time division TD_n ends and time division TD_{n+1} begins with the transmission of sync pulse 202 over bus 104. Time division TD_{n+1} ends and time division TD_{n+2} begins with the transmission of sync pulse 203 over bus 104. The sync pulses

5   occur periodically to produce equal time divisions. Any node on the network can be configured to generate the sync pulses. One node is simply designated as the "master node" by assigning the master node the lowest address used in the network, e.g. zero. A node address of zero means messages from the master node have the highest priority. In a preferred embodiment of the present invention, a

10   sync pulse is transmitted over bus 104 every 10.0 milliseconds producing 100 time divisions per second. Those skilled in the art recognize that the time division of a bus system can be tuned depending on the configuration of the system. The 10.0 millisecond time division of the preferred embodiment of the present invention is merely suitable for a particular bus system configuration. Each of messages 205-

15   215 is a message transmitted over bus 104. The format and bit designation of each message is discussed above with respect to FIG. 1 and existing CAN systems. For simplicity each message in FIG. 2 is represented as a single box along timeline 100 rather than showing the more detailed bit by bit representation.

An industrial process control example is provided to further explain the CAN-

20   D system. Node 101 in FIG. 1 includes a mass flow sensor (not shown). Node 101 is configured to produce, during each time division, a message containing the current mass flow rate in grams per second of a fluid flowing through the mass flow sensor. Messages produced by Node A are real-time messages. Message 205 is transmitted by Node 101 during time division TD_n, message 209 is transmitted by

25   Node 101 during time division TD_{n+1} and message 213 is transmitted by Node 101 during time division TD_{n+2}. Node 102 includes a pressure sensor. Node 102 is configured to produce, during each time division, a message containing the current drop in fluid pressure across the mass flow sensor in pounds per square inch. Messages produced by Node 102 are real-time messages. Message 206 is

30   transmitted by Node 102 during time division TD_n, message 210 is transmitted by Node 102 during time division TD_{n+1} and message 214 is transmitted by Node 102 during time division TD_{n+2}. The length of the periodic time divisions, the total

11

number of nodes, the number of real-time nodes, and the length of each message are controlled by the system design to ensure that there is enough time during each time division for the necessary communications to occur.

During each time division, the mass flow rate message from node 101 and
5   the pressure drop message from node 102 are read by node 103 and processed by microprocessor 116. Microprocessor 116 combines the data from nodes 101-102 in a known way to produce a value representative of the viscosity of the measured fluid. The deterministic characteristic of the CAN-D system ensures that microprocessor 116 calculates viscosity using mass flow rate data and pressure
10   data generated at or about the same time. In the case of the embodiment where the time divisions are 10.0 milliseconds long, one knows that the mass flow rate data and the pressure data were generated within 10.0 milliseconds of one another.

Viscosity data produced by microprocessor 116 may be used in any of several different ways. For example, viscosity data may be presented on a display
15   (not shown) for viewing by a user. Alternatively, viscosity data may be transmitted over bus 104 to another node on bus 104.

Non-real-time messages are transmitted over bus 104 whenever they are available and priority arbitration permits their transmission. Messages 207, 208, 211, 212 and 215 are additional messages transmitted over bus 104. These
20   messages may be real-time or non-real-time messages.

Non-real-time messages are not necessarily low priority as compared to real-time messages. Using the above example, an additional node (not shown) includes a level sensor producing a level alarm. The level sensor is mounted in a tank into which is flowing the fluid whose viscosity is being measured. When the fluid
25   reaches a certain level in the tank, the level sensor produces a level alarm which is transmitted as a message over bus 104. The level alarm message is not a real-time message but it is assigned a relatively high priority since failure to respond to the level alarm could cause an overflow of the tank.

The CAN-D bus communication system described with respect to FIG. 2
30   utilizes the message formatting, priority arbitration and error checking described with respect to the CAN protocol and FIG. 1. In addition, CAN-D makes use of commercially available CAN controller and transceiver chips. The sync message

used to create a system time in CAN-D imbues the standard CAN protocol with a
TDMA-like determinism. Thus, CAN-D is particularly well suited to real-time
process control applications or any other application requiring the transmission of
real-time data.

5   The above example where Node 101 is a mass flowmeter and Node 102 is
a pressure transducer is an example of the CAN-D system applied to a
communications bus 104 between separate instruments in a process plant. One
skilled in the art recognizes that the same principles of the CAN-D system are
applicable to communications between two nodes of any type.

10   CAN-D Node Activity - FIG. 3

        FIG. 3 is a family of timing diagrams 200, 300 and 301 illustrating, in general,
the operation of the present invention including message-generation activity within
nodes 101-102. Timeline 200 is identical to timeline 200 of FIG. 2 and indicates
bus activity (CAN messages) over bus 104. Timelines 300 and 301 illustrate
15   activity within nodes 101-102, respectively, during each time division. Whereas
each box along timeline 200 denotes a CAN message over bus 104, each triangle
along timelines 300 and 301 denotes the generation of a message content within
nodes 101-102, respectively. FIG. 3 illustrates the timing relationships between the
generation of message information within nodes and the transmission of those
20   messages over bus 104.

        Timeline 300 represents activity within node 101 of FIG. 1. Likewise,
timeline 301 represents activity within node 102 of FIG. 1. For purposes of
describing the overall operation of the CAN-D system it is not necessary to explain
or understand the processing occurring within each node. It is only necessary to
25   understand that standard CAN formatting and control logic is used at each node to
make available messages for transmission over bus 104. For example, a node
may be configured such that each sync pulse 201-204 causes the node to initiate
the generation of a new message. Alternatively, a node may be configured such
that every third synch pulse causes the node to initiate the generation of a new
30   message. It is not necessary that each node, real-time or non-real-time, produce
a message during each time division. Also, a single node might produce more than
one message during a single time division. Each sync pulse is transmitted to every

13

node on a CAN-D network thereby providing a common sense of time to every node.

CAN message 209 transmitted over bus 104 during time division TD_{n+1} contains message content 302. Message content 302, illustrated on timeline 300,

5   is generated by microprocessor 107 during time division TD_n. The generation of message content 302 occurs because node 101 is configured to produce a real-time message during every time division. Alternatively, message content 302 can be generated in response to a "publication demand" requesting the generation of the message. A publication demand instructs a node to prepare a message for

10   transmission over the bus. Publication demands can originate from a node over bus 104 as a system message or can originate within each node. Message content 302 is queued within node 101 until sync pulse 202. Message content 304 is generated within node 102 during time division TD_n and is queued within node 102 until sync pulse 202.

15   After each sync pulse 201-204 occurs, the messages queued at the nodes are transmitted over bus 104 and new messages are generated and queued at the nodes. For example, after synch pulse 202, the messages queued at nodes 101-102 are transmitted over bus 104 and new messages are generated and queued at nodes 101-102. Message content 302, generated and queued within node 101

20   during time division TD_n, is transmitted over bus 104 as message 209 after sync pulse 202 during time division TD_{n+1}. Message content 304, generated and queued within node 102 during time division TD_n, is transmitted over bus 104 as message 210 after sync pulse 202 during time division TD_{n+1}. Messages from node 101 have a higher CAN priority than messages from node 102 therefore message 209 is

25   transmitted over bus 104 before message 210 is transmitted over bus 104.

Message content 303 is generated within node 101 and message content 305 is generated within node 102 during time division TD_{n+1}. Both messages are queued within their respective nodes until the next sync pulse 203. As occurs during each time division, queued messages are transmitted over bus 104 after

30   each synch pulse. After sync pulse 203, message content 303 is transmitted over bus 104 as message 213 and message content 305 is transmitted over bus 104 as message 214. It does not matter when during a time division a certain message

content is generated. It only matters that a real-time message is generated during each time division, if the node is so configured. Standard CAN priority arbitration will ensure the proper order of transmission of messages over bus 104.

Messages 205 and 206 transmitted over bus 104 during time division TD_n
5   were generated during a previous time division TD_{n-1} (not shown). Sync pulse 201 initiated their transmission from nodes 101-102, respectively, over bus 104. The above discussion of FIG. 3 relates to real-time messages transmitted over bus 104. A deterministic communications bus requires the ensured schedulability of the actions and messages in a real-time system. The CAN-D system ensures that real-
10 time data will be communicated during known and predictable time divisions. The deterministic quality of the CAN-D system is not important to the transmission of non-real-time messages. During each time division, the real-time messages are transmitted over the CAN bus before the non-real-time messages. This is guaranteed by assigning higher priority CAN addresses to the real-time nodes than
15 are assigned to the non-real-time nodes. After the transmission of all real-time messages, the time remaining in a given time division is available for the transmission of non-real-time messages. If any non-real-time messages are not sent because the time division ends, the remaining non-real-time messages are held over for transmission during the next time division. During the next time
20 division, transmission of the held-over non-real-time messages again must wait until the real-time messages are sent.

CAN-D Segmented Time Divisions - FIG. 4

Various implementations of CAN-D are possible that all result in a deterministic, CAN communications environment. FIG. 4 illustrates a timing
25 diagram of one embodiment of the present invention where each time division is segmented into three segments and the sync signal performs multiple functions in addition to signaling the start of a new time division.

Timeline 400 illustrates message activity over bus 104 during two time divisions. Time division TD_n, which is representative of each time segment, is
30 broken into three segments 401-403. Segment 401 is the "sync segment", segment 402 is the "real-time segment" and segment 403 is the "non-real-time segment". Time division TD_n is initiated by the transmission of sync message 407. Sync

message 407 is a high priority system message that is transmitted for the duration of sync period 401. Sync message 407 is, like the sync pulse of FIGS. 2-3, a CAN message having the highest priority.   Transmission of sync message 407 accomplishes several objectives. One objective accomplished by the transmission

5   of sync message 407 is the initiation of a new time division. Another objective accomplished by the transmission of sync message 407 is that the transmission of all other messages are held off the bus due to the high priority of the sync message. A further effect of the sync message is as a signal or interrupt to each node to build its real-time messages. Messages are prepared at each node during

10   synch period 401 and their transmission over bus 104 awaits standard CAN arbitration during segments 402-403.   The duration of the sync message is the same length of time during each time division. In one embodiment using 10.0 millisecond time divisions, for example, the sync message lasts for 1 millisecond.

During segment 401, which corresponds to transmission of sync message

15   407, the various real-time producing nodes generate and queue their respective real-time messages. Segment 402 of TD_n begins when sync message 407 is complete. Standard CAN priority arbitration then automatically sorts the queued real-time messages and allows them to be sent out over bus 104 during segment 402. The real-time and non-real-time messages are prioritized such that all the

20   real-time messages, but only the real-time messages, are transmitted during the real-time segment of each time division.

Segment 403 of TD_n begins when all the real-time messages have been transmitted over bus 104. The number of real-time nodes connected to bus 104 must be selected such that each one can transmit a message during each real-time

25   segment and still leave a long enough non-real-time segment for a sufficient amount of non-real-time messages. Any non-real-time messages for which there is not enough time to transmit during a given time division are held off the bus until the non-real-time segment of the next time division.

Time division TD_{n+1} is segmented into sync segment 404, real-time segment

30   405 and non-real-time segment 406. CAN messages are formed and transmitted during time division TD_{n+1}, and other time divisions (not shown), as described with respect to time division TD_n.

Intra-Nodal Communications -FIG. 5

The communications protocol of the present invention is also applicable to communications between functional blocks within a single node. FIG. 5 illustrates a node 500 connected to a CAN-D bus 501. Node 500 includes the major
5   components discussed with respect to FIGS. 1-4 namely, CAN transceiver 502, CAN controller 504 and microprocessor 506. CAN transceiver 502 controls the physical interface between node 500 and CAN-D bus 501. CAN transceiver 502 communicates over path 503 to CAN controller 504 which implements the standard CAN protocol features such as priority arbitration and error checking. CAN
10   controller 504 communicates over path 505 to microprocessor 506. Microprocessor 506 is shown in greater detail than the microprocessors of FIG. 1. Programmed instructions for the operations performed by node 500 are executable on Central Processing Unit ("CPU") 507. CPU 507 stores and/or retrieves the programmed instructions and/or data from memory devices that include, but are not limited to,
15   Random Access Memory (RAM) 509 and Read Only Memory (ROM) 510 by way of memory bus 508. Another accessible memory device includes non-volatile memory device 512 by way of local bus 511. Programmed instructions for implementing multiple applications 513-515 are stored on non-volatile memory device 512. Software router 516 directs the intra-nodal operation of node 500, as
20   described below. Various input and output devices (not shown) are connected to local bus 511 to allow a user to interact with the functionality of node 500.

Software router 516 is an application executed by CPU 507. Software router 516 is programmed by a user to direct the operation of other applications or functional blocks within node 500. For example, in one embodiment of node 500,
25   a measurement of viscosity is delivered by node 500 to CAN-D bus 501 every other time division. Software router 516, as executed by CPU 507, responds to each sync signal to direct operations by applications 513-515. In the current example, application 513 produces a mass flow rate datum, application 514 produces a differential pressure datum and application 515 produces a viscosity datum.
30   Application 515 uses the data from applications 513-514 to produce the viscosity datum. Applications 513-515 produce their respective output when directed to do so by software router 516. Software router 516 directs which of applications 513-

515, if any, should produce output subsequent to each sync signal. For the current example, software router 516 is programmed to 1) request application 513 (mass flow rate) produce output during each time division, 2) request application 514 (differential pressure) produce output during alternating time divisions, and 3)

5   request application 515 (viscosity) produce output during alternating time divisions (the time divisions when application 514 is not producing output). In response to a first sync signal, software router 516 instructs application 513 and 514 to produce output. During a first time division (initiated by the first sync signal), applications 513 and 514 produce output messages to software router 516. In response to a

10   second sync signal, software router 516 directs the output messages from applications 513-514 (generated during the previous time division) to application 515 and directs the mass flow rate output message from application 513 to CAN-D bus 501. Other nodes (not shown) on CAN-D bus 501 utilize the mass flow rate output from node 500. Software router 516 also requests application 515 to

15   produce a viscosity output message. Application 515 processes the mass flow rate and differential pressure messages produced by applications 513 and 514 (generated during the previous time division), respectively, to produce a viscosity message which is sent to software router 516. During a third time division (after a third sync signal), software router 516 directs the viscosity message (received from

20   application 515 during the previous time division) to CAN-D bus 501 for consumption by other nodes (not shown).

The CAN-D protocol and software router allow for flexible implementation and amendment of an overall system architecture. For example, it may be more efficient to have the mass flow rate information generated in a node different and

25   separate from the viscosity-producing node. Application 513 is thus moved from node 500 to a different node (not shown) on CAN-D bus 501. No changes need be made to application 513. It simply resides on another node where that node's software router instructs application 513 when to produce output.

A "software router" is not required according to the communications protocol

30   of the present invention. For example, each functional block within a node could instead be configured to produce the appropriate message(s) during each time division and send the message(s) so they are received during an appropriate time

18

division. The software router simply serves as a convenient scheduling tool which, among other things, queues a message received during one time division and sends the queued message to the appropriate functional block during the next time division. This ensures that messages are communicated in a fashion that

5　preserves the deterministic quality of the communications protocol of the present invention.

In a further embodiment of the present invention, the intra-nodal communication described with respect to FIG. 5 is utilized where a node does not exchange data with a CAN-D bus or any other type of bus. With reference to FIG.

10　5, only microprocessor 506 is relevant to this embodiment and the remaining elements, e.g. CAN transceiver 502, CAN controller 504 and CAN-D bus 501, either do not exist or are not utilized. Rather than receiving a sync pulse from a master node on a bus, a timer is implemented utilizing operating system methods to produce a periodic event having the equivalent effect as a sync pulse. An

15　operating system is a set of services which provide a means by which 1 or more functional blocks can share a hardware platform. Each timer event marks the start of a new time division during which software router 516 sends request messages to functional blocks 513-515. Functional blocks 513-515 receive the request messages from software router 516 and deliver the requested data back to

20　software router 516. In this embodiment, software router 516 takes advantage of a message queue provided by the operating system to queue the messages from functional blocks 513-515 until the next time division. At the start of the next time division, software router 516 distributes the queued messages to the appropriate ones of functional blocks 513-515 and sends the next set of request messages to

25　functional blocks 513-515. Messages are prioritized in this embodiment according to operating system methods by assigning a high priority to real-time messages and lower priority to non-real-time messages. This embodiment allows for deterministic, real-time processing within a single node even when the node is not connected to a CAN-D bus. Of course, a node operating in this fashion is easily integrated into

30　a CAN-D bus environment, if appropriate.

Nodal Processing Steps - FIG. 6

      FIG. 6 depicts the steps which occur at each node in response to a sync signal. FIG. 6 is used to describe the operation of the embodiment including multiple nodes on a CAN-D bus as well as the embodiment of a single node not

5   connected to a CAN-D bus but communicating among multiple functional blocks according to a CAN-D-based protocol. During step 600 the sync signal is received at a node. As noted above, the sync signal is transmitted from a master node to the bus or, alternatively, may be generated as a local timer event in the case where a node operates independently from a bus. A common sense of time among all the

10   nodes of the bus and/or all the functional blocks within a node is achieved by each node resetting a new time division with each sync signal. A new time division begins with each sync signal and processing continues to step 601.

      During step 601 any real-time messages that were queued during the previous time division are appropriately communicated. In the case of multiple

15   nodes connected to a CAN-D bus, the real-time messages are put on the CAN-D bus. In the case of a signal node, the real-time messages are distributed to the appropriate functional blocks. Processing then continues to step 602.

      During step 602 real-time data for the current time division is generated. Depending on the configuration of the node (and/or of the software router in the

20   node) some real-time data may be sent during the time division in which the real-time data is generated. Non-real-time data is typically sent whenever there is time available during a time division. This may be after all the real-time data has been sent or during a period when real-time data is being generated and the bus is idle. Processing continues from step 602 to decision block 603 where, if more time is

25   available in the current time division, processing is passed to step 604. During step 604 non-real-time data is sent. Real-time data may also be sent during this step. Steps 603-604 are repeated until another sync signal is generated indicating the start of a new time division.

      Processing steps 600-604 operate to transmit real-time data first during each

30   time division. The remaining time is used for the transmission of non-real-time data.

Summary

The serial bus communication system of the present invention combines the simplicity and availability of low-cost CAN components and protocol with the attribute of determinism. Thus the deterministic-CAN protocol of the present invention is well-suited for the communication of real-time data as compared to prior CAN implementations.

WE CLAIM:

1.   A method of deterministically communicating data between multiple nodes (500) consistent with the Controller Area Network ("CAN") serial communications protocol, comprising the steps of:

        providing a communication means (501) between said multiple nodes

5 wherein each of said multiple nodes is available to said communications means;

        generating a periodic event (401) recognized over said communication means by said multiple nodes;

        initiating, responsive to said periodic event, common time divisions (TD_n) at said multiple nodes; and

10   transmitting said data between two nodes of said multiple nodes over said communication means wherein said data is transmitted during a known one of said time divisions.

2.   The method of claim 1 wherein said providing step includes:

        providing a communication means wherein said communication means is a data bus to which all of said multiple nodes are connected.

3.   The method of claim 2 wherein said generating step includes:

        generating a periodic event wherein said periodic event is a periodic synchronization signal; and

        issuing said periodic synchronization signal from one of said multiple nodes

5 over said data bus to said multiple nodes.

4.   The method of claim 2 wherein said providing step includes:

        translating messages generated by each of said nodes to messages consistent with said CAN protocol.

5.   The method of claim 2 wherein said step of providing said data bus includes:

        connecting each of said multiple nodes to said data bus wherein each of said nodes includes an independent instrument.

6.   The method of claim 5 wherein said connecting step includes:

translating messages generated at each of said nodes with a CAN controller to provide messages from each of said nodes that are consistent with said CAN protocol.

7.   The method of claim 2 wherein said step of providing said data bus includes:

connecting each of said multiple nodes to said data bus wherein each of said nodes includes a block of functionality for a single instrument.

8.   The method of claim 7 wherein said connecting step includes:

translating messages generated at each of said nodes with a CAN controller to provide messages from each of said nodes that are consistent with said CAN protocol.

9.   The method of claim 3 wherein said issuing step includes:

generating at a master node which is one of said nodes a high priority CAN message as said synchronization signal; and

issuing said synchronization signal periodically over said data bus to each
5   of said nodes.

10.   The method of claim 9 wherein said generating step includes:

generating at said master node a synchronization signal comprised of multiple, successive high priority CAN messages; and

issuing said synchronization signal periodically over said data bus to each
5   of said nodes.

11.   The method of claim 3 wherein said issuing step includes:

receiving said periodic synchronization signal at said multiple nodes; and

resetting a local time division at each of said nodes such that each of said nodes operates according to said common time divisions.

12.   The method of claim 11 wherein said issuing step further includes:

forming, responsive to said resetting step, a CAN message for transmission over said data bus.

13.   The method of claim 12 wherein said forming step includes:

forming, responsive to said resetting step, a CAN message containing real-time data for transmission over said data bus.

14.   The method of claim 11 wherein said issuing step further includes:

segmenting each of said time divisions into segments during which different classes of said data are transmitted over said data bus.

15.   The method of claim 1 wherein said providing step includes:

providing a communication means between said multiple nodes wherein each of said multiple nodes is a block of functionality within a single instrument.

16.   The method of claim 15 wherein said providing step includes:

providing a communication means between said multiple nodes wherein said communication means is an operating system of said instrument.

17.   The method of claim 16 wherein said generating step includes:

generating a periodic event wherein said periodic event is a timer output of said operating system; and

passing said timer output from said operating system to said multiple nodes.

18.   An apparatus for deterministically communicating data between multiple nodes (500) according to the Controller Area Network ("CAN") serial communications protocol, comprising:

a data bus (501) to which is connected each of said multiple nodes;

5   a master node which is one of said nodes for issuing a periodic synchronization signal (401, 411) over said data bus to said multiple nodes;

means (516) at each of said nodes for initiating, responsive to receipt of said periodic synchronization signal, a time division; and

means (502, 504) for transmitting said data between two nodes of said
10 multiple nodes over said data bus wherein said data is transmitted during a known said time division.

19.   The apparatus of claim 18 wherein said data bus includes:

means at each of said multiple nodes for translating messages generated within said nodes to a format consistent with said CAN protocol.

20.   The apparatus of claim 19 wherein said translating means includes:

physical interface means at each of said nodes for interfacing said data bus and each of said nodes; and

CAN protocol means for translating messages generated within said nodes
5 to a format consistent with said CAN protocol.

21.   The apparatus of claim 18 wherein said master node includes:

timing means for establishing the timing of said periodic synchronization signal.

22.   The apparatus of claim 18 wherein said initiating means includes:

means, responsive to said synchronization signal, for generating a CAN message containing said data; and
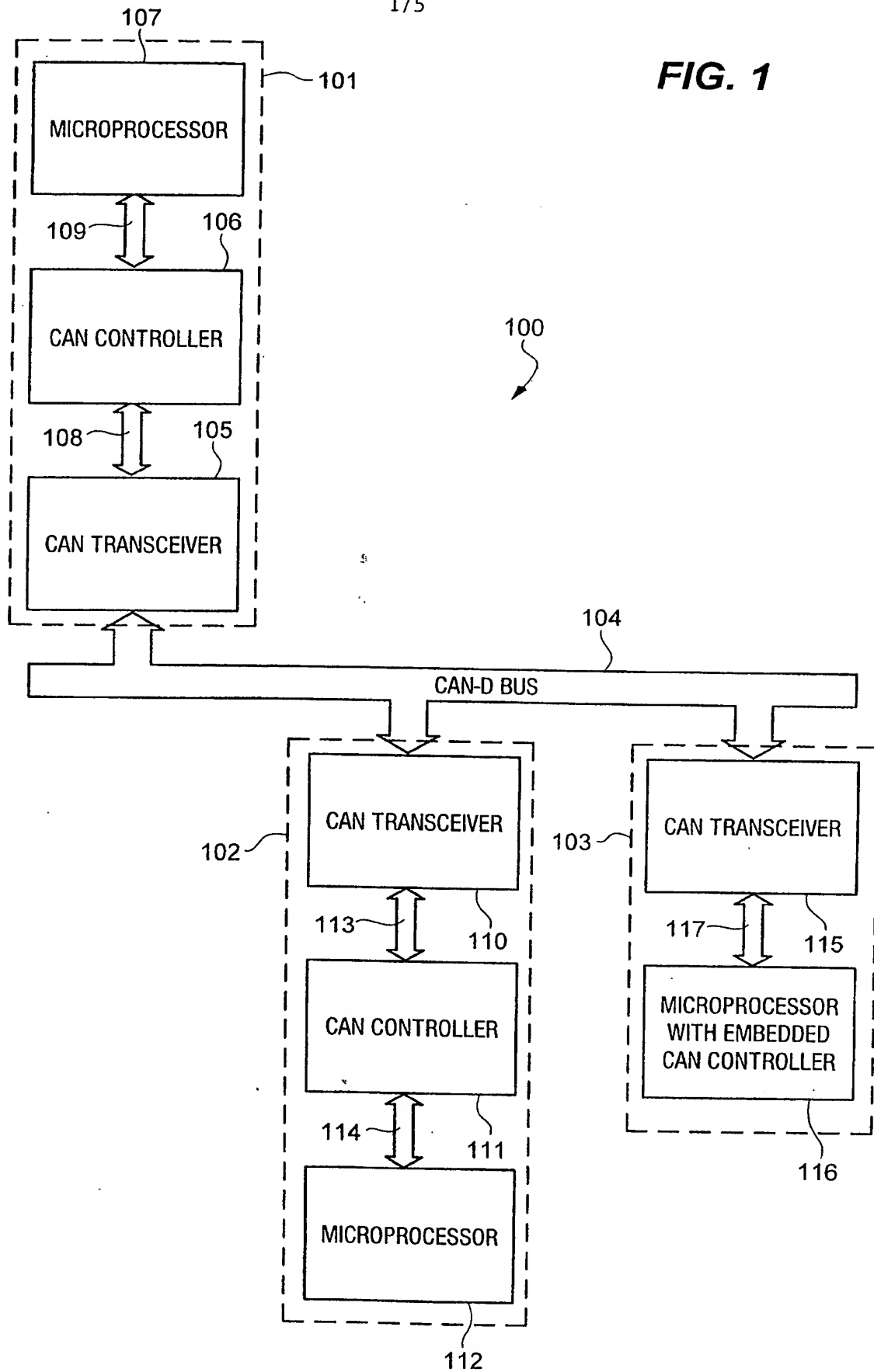
means, responsive to said generating means, for requesting to send said
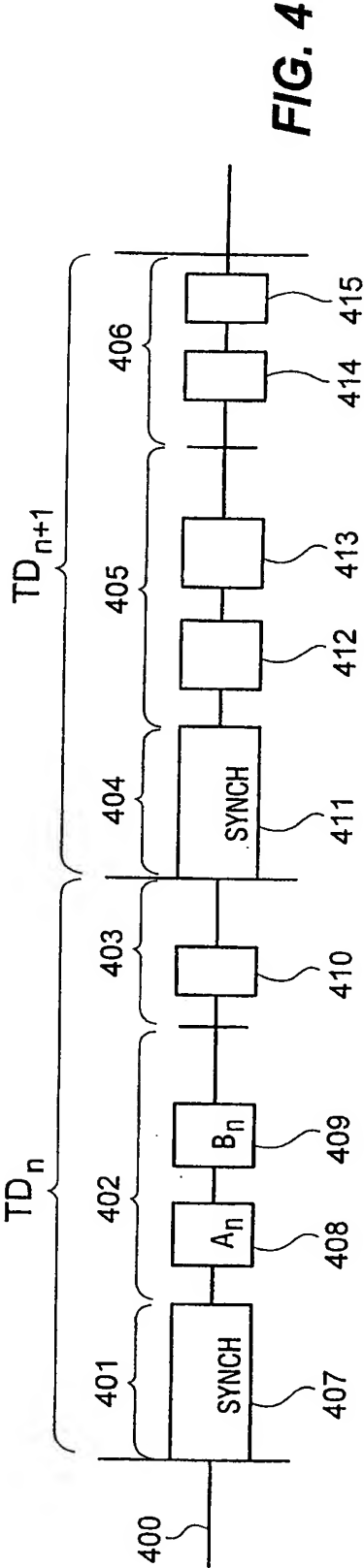5 CAN message over said data bus.

23.   The apparatus of claim 22 wherein said transmitting means includes:

means, responsive to said requesting means, for arbitrating which said CAN message of multiple CAN messages is the next message to transmit over said data bus; and

5   means, responsive to said arbitration means, for sending said CAN message
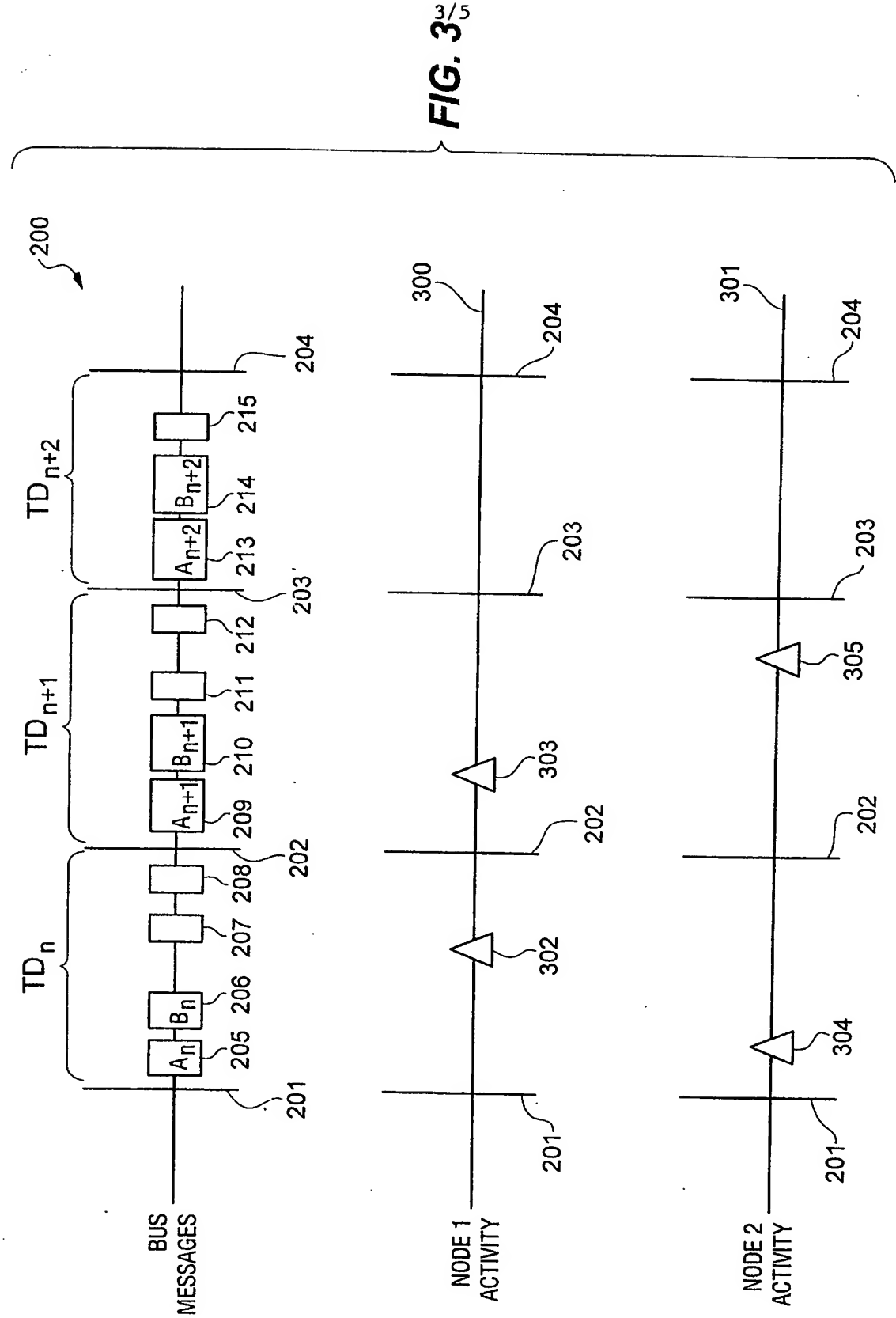from one of said nodes to said data bus from where said CAN message is
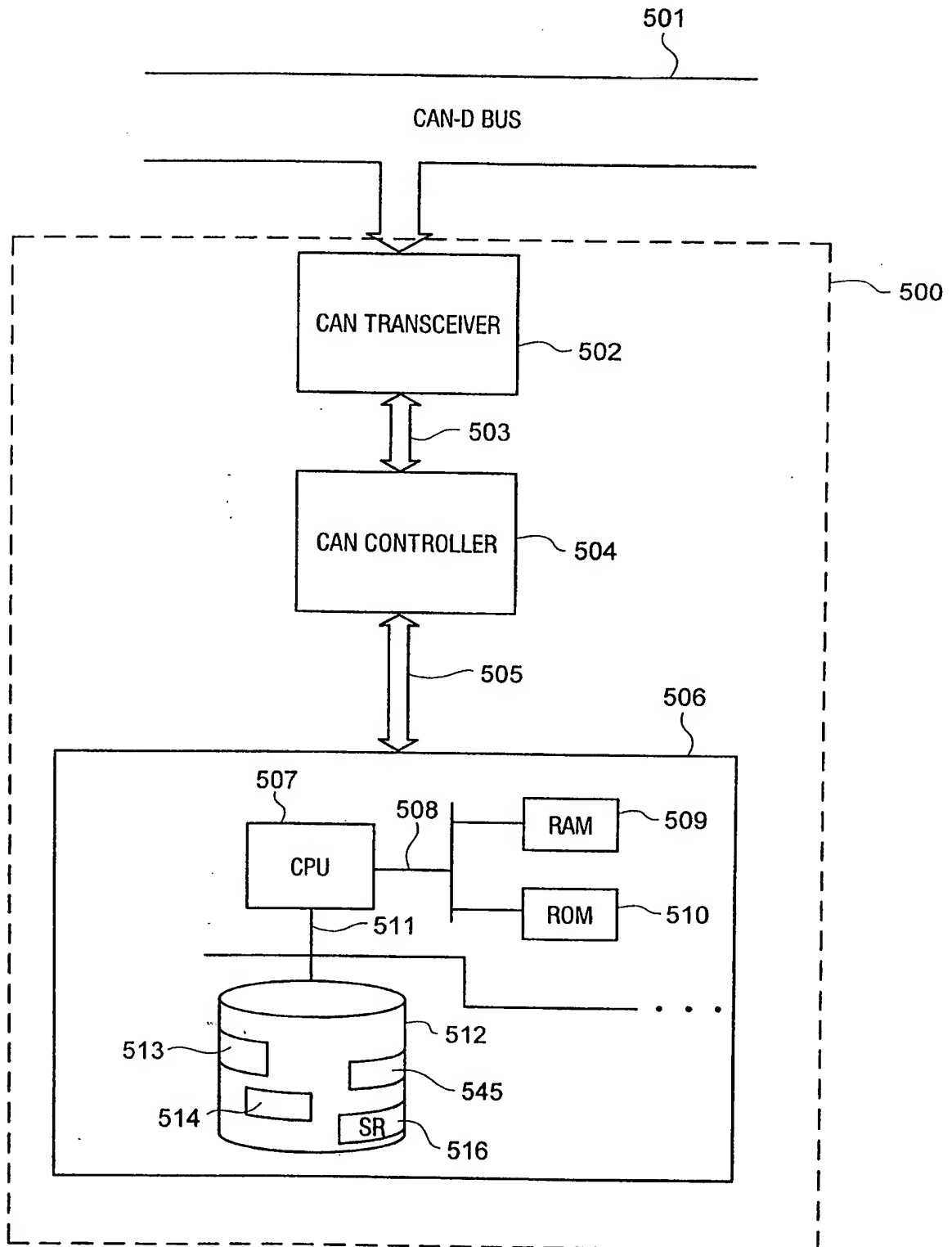accessible by others of said multiple nodes.

FIG. 1
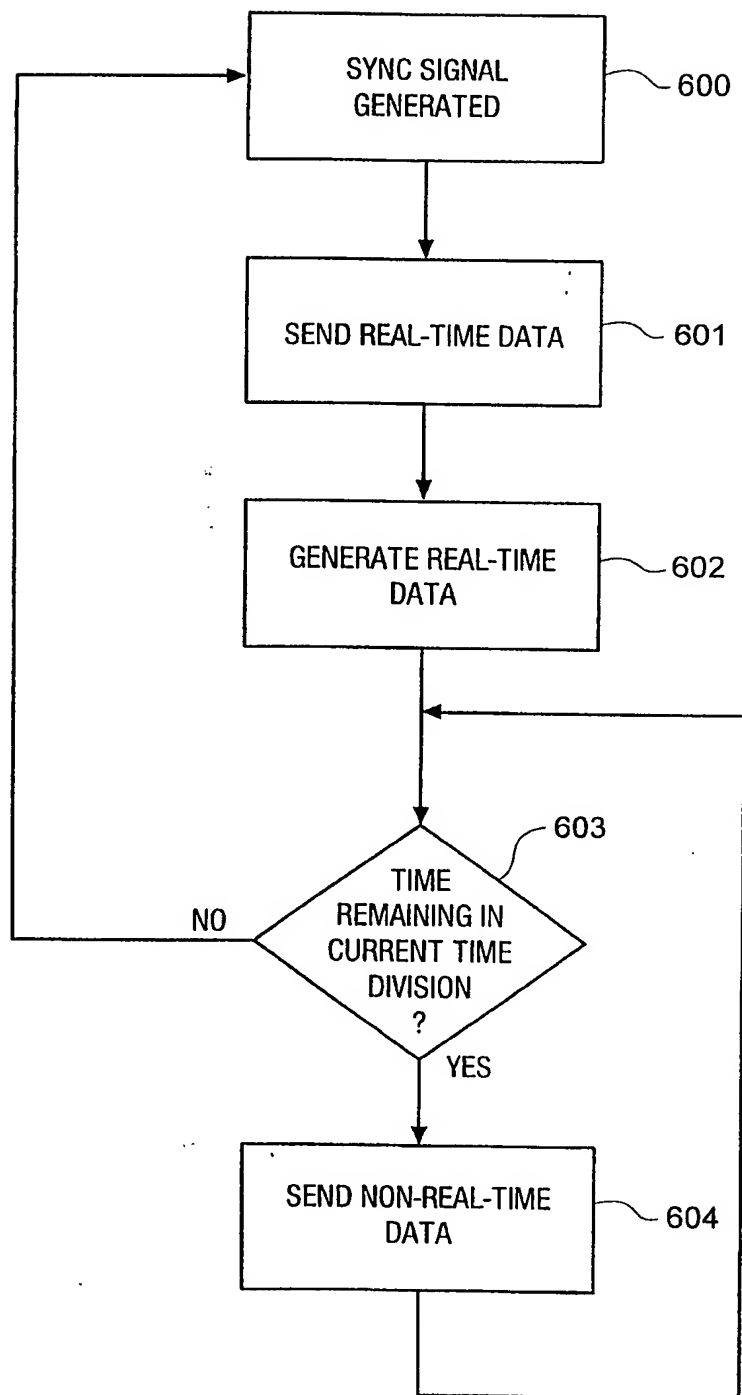
FIG. 2

FIG. 4

FIG. 3

FIG. 5

FIG. 6

```
        ┌──────────────────────┐
   ┌───▶│    SYNC SIGNAL       │──── 600
   │    │     GENERATED        │
   │    └──────────────────────┘
   │               │
   │               ▼
   │    ┌──────────────────────┐
   │    │  SEND REAL-TIME DATA │──── 601
   │    └──────────────────────┘
   │               │
   │               ▼
   │    ┌──────────────────────┐
   │    │  GENERATE REAL-TIME  │──── 602
   │    │        DATA          │
   │    └──────────────────────┘
   │               │
   │               ▼
   │            ◇ 603
   │          ╱       ╲
   │        ╱   TIME    ╲
   │  NO  ╱  REMAINING IN ╲
   └─────  CURRENT TIME   ◀──────┐
         ╲   DIVISION    ╱       │
          ╲     ?      ╱         │
            ╲       ╱            │
               │ YES             │
               ▼                 │
        ┌──────────────────────┐ │
        │ SEND NON-REAL-TIME   │ 604
        │       DATA           │ │
        └──────────────────────┘ │
               │                 │
               └─────────────────┘
```

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6   H04L12/413   H04L12/417

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6   H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and. where practical. search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document. with indication. where appropriate. of the relevant passages	Relevant to claim No.
X	J.-D. DECOTIGNIE ET AL.: "PRODUCER-DISTRIBUTOR-CONSUMER MODEL ON CONTROLLER AREA NETWORK" 20TH IFAC/IFIP WORKSHOP ON REAL-TIME PROGRAMMING, 6 - 10 November 1995, FORT LAUDERDALE, FLORIDA, US, pages 35-42, XP002074139 see page 39, right-hand column, line 5 - page 42, right-hand column, line 11; figures 6,7	1-23
A	DE 44 08 488 A (BOSCH GMBH ROBERT) 21 September 1995 see column 7, line 33 - column 8, line 61; claim 1	1,18

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 August 1998	28/08/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Messelken, M

Form PCT/ISA/210 (second sheet) (July 1992)

2

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
DE 4408488 A	21-09-1995	WO 9525024 A	21-09-1995
		DE 59500287 D	10-07-1997
		EP 0701515 A	20-03-1996
		JP 8510625 T	05-11-1996
		US 5748923 A	05-05-1998